The WIG Project

Matthew Dwyer
324E Nichols Hall
dwyer@cis.ksu.edu
http://www.cis.ksu.edu/~dwyer
Course Project:

• **Web Interface Generator**;
• compiled into C-based CGI-scripts;
• illustrates a domain-specific language;
• shows server-side web programming;
• used to get hands-on experience;
• and you will implement the language from scratch in C or Java.
Uses of WWW

• Static documents
  – Supported by HTML

• Dynamic documents
  – Supported by HTML and CGI

• Interactive services
  – Supported by CGI, …

• Distributed programming
  – Aglets, …
Web Content

• Static documents
  – lots of individual files needed
  – the documents are rarely updated; and
  – the documents are not customized.

• Dynamic documents:
  – there are fewer documents;
  – the documents are always updated; and
  – the documents are customized.
Standard Web Interaction

Client

Server

HTML
Common Gateway Interface

Client

Server

- fill-out form
- form data
- dynamic document
- HTML
- script
Fill-out Forms

• An extension to HTML
• The `<form ...>` tag contains:
  – transmission method (**POST** or **GET**);
  – URL of the script; and
  – a query string.
• Extra tags for input fields:
  – simple text fields;
  – radio buttons;
  – menus; and
  – submit buttons.
A Simple Fill-out Form

Your name: [ ]

Your quest: to find the Holy Grail [ ]

Your favorite color: [ ] red [ ] green [ ] blue [ ] I don’t know

Answer
<form>
  method="POST"
  action="http://www.brics.dk/cgi-mis/Python?Questions"
>
Your name:
<input name="name" type="text" size=20>.
<p>
Your quest:
<select name="quest">
<option value="grail">to find the Holy Grail
<option value="wig">to write a WIG compiler
</select>
Your favorite color:
<input name="color" type="radio" value="red">red
<input name="color" type="radio" value="green">green
<input name="color" type="radio" value="blue">blue
<input name="color" type="radio" value="argh">I don't know

<input name="submit" type="submit" value="Answer">
</form>
CGI Script

• The script may be written in any programming or scripting language.

• The form data appears on standard input as:
  
  name=Michael&quest=wig&color=blue&submit=Answer

• but must first be decoded:
  
  – change '+' into a space character; and
  
  – replace \%xy by the ASCII character with hex value xy

• Reserved symbols in data, e.g., '=' and '&', must be encoded.
CIS 706 Translators I

Script Environment Variables

SERVER_SOFTWARE = NCSA/1.5.2
SERVER_NAME = www,cis.ksu.edu
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = POST
SCRIPT_NAME = /cgi-bin/python
QUERY_STRING = Questions
REMOTE_HOST = sarin
REMOTE_ADDR = 130.225.16.12
REMOTE_USER =
CONTENT_TYPE = application/x-www-form-urlencoded
CONTENT_LENGTH = 47
USER = dwyer
Dynamic Document

- Scripts produce dynamic web-page on standard output
  
  Content-type: text/html
  
  Hello Matt,
  
  <p>
  Good luck on writing a green WIG compiler!
  
  - Or redirect pages from an existing file
  
  Location: URL
  Content-type: text/html
CGI Protocol is State-less

- Client-server (script) exchanges
  - Happen in isolation;
  - No information remains on the server;
  - Different users cannot communicate through a service.

- We would like to program services with
  - Global state;
  - Per user sessions;
  - Threads; and
  - Local state.
Guessing Game

Please guess a number between 0 and 99: 50

That is not correct. Try a higher number: 75

That is not correct. Try a higher number: 87

That is not correct. Try a higher number: 93
Guessing Game

That is not correct. Try a higher number: 97

You got it, using 6 guesses.

That is not correct. Try a lower number: 95

Thanks for playing this exciting game.

Session Terminated: posewybcwbbjjxjdypya
WIG

• Language for developing interactive services that provides
  – Global state;
  – Safe, dynamic documents;
  – Sequential sessions;
  – Multiple threads;
  – Local state within a session; and
  – Simple concurrency control.

• WIG programs are compiled to CGI-scripts
Nikolaj’s Counter

service {
  global counter: int = 0;

  document Nikolaj {
    <img src="http://www.brics.dk/~mis/babybath.jpg">
    <p>
      <i>You are visitor number <var name="no"></i>
    }

  session Access {
    counter:=counter+1;
    show Nikolaj[no<-counter]}
}

CIS 706 Translators I
Generated Web-page
service {
    global plays: int;
    global record: int;
    global holder: text;

    document Init { ... }
    document Retry { ... }
    document Again { ... }
    document Done { ... }
    document Record { ... }
    document Finish { ... }
    document List { ... }

    session Play { ... }
    session HiScore { ... }
}
session Play {
    local number: int;
    local guesses: int;
    local guess: int;
    number := random(100);
    plays := plays + 1;
    guesses := 1;
    show Init[guess->guess];
    while guess > 99 { show Retry[guess->guess] };
while not (guess=number) {
    guesses:=guesses+1;
    if guess>number {
        show Again[correction="lower",guess->guess]
    } else {
        show Again[correction="higher",guess->guess]
    }
    while guess>99 { show Retry[guess->guess] }
};
show Done[trys<-guesses];
    if (record = 0) or (record > guesses) {
        show Record[old<-record,name->holder];
        record:=guesses
    };
    show Finish
}
document Init {
    Please guess a number between 0 and 99:
    <input name="guess" type="text" size=2>
}

document Retry {
    That number is too large!
    <p>
    Please keep your guess between 0 and 99:
    <input name="guess" type="text" size=2>
}
document Again { 
    That is not correct. 
    Try a <var name="correction"> number: 
    <input name="guess" type="text" size=2>
}

document Done { 
    You got it, using <var name="trys"> guesses. 
}
document Record {
    That makes you the new record holder, beating the old record of \(<\text{var name="old"}>guesses</\text{var}>\).
    <p>Please enter your name for the hi-score list <input name="name" type="text" size=20> </p>
}

document Finish {
    Thanks for playing this exciting game.
}
documents : document | documents document ;
document : "document" identifier title "{" html "}" ;
title : /* empty */ | "(" string ")" ;
html : entity | html entity ;
entity : literal
    | "<b>" html "</b>"
    | "<i>" html "</i>"
    | "<h1>" html "</h1>"
    | "<h2>" html "</h2>"
    | "<p>"
    | "<br>"
    | "<hr>“
    | ...
entity : …
| "<img" "src" "=" string "">
| "<include" "file" "=" string "">
| "<a" "href" "=" string ">" html "</a>"
| "<ul" items "</ul>"
| "<table" rows "</table>"
| "<var" "name" "=" string "">
| "<select" "name" "=" string "">" options "</select>"
| "<input" "name" "=" string "type" "=" """"text"" "=" """"size" "=" "intconst "">
| "<input" "name" "=" string "type" "=" """"radio"" "=" "value" "=" "string "">
| "<applet" "codebase" "=" string "code" "=" string
"width" "=" intconst "height" "=" intconst "">
arguments "</applet>"
WIG Documents

• Every document is implicitly a form;
• The `<var ...>` tag defines _holes_ to be filled in dynamically; and
• The `<include ...>` tag allows static sub-documents
WIG Statements

statement : /* empty */
| "stop"
| identifier ":=" exp
| "take" identifier "{" statement "}"
| statement ";" statement
| "show" identifier parameters
| "if" exp "{" statement "}"
| "if" exp "{" statement "}" "else" "{" statement "}"
| "while" exp "{" statement "}"

parameters : /* empty */ | "[" "]" | "[" neparlist "]"

neparlist : parameter | neparlist "," parameter

parameter : identifier "<-" exp
| identifier "->" identifier
Comments on WIG Statements

• there are write locks on global variables;
• the \texttt{take} statement defines a critical region; and
• the \texttt{show} statement displays a dynamic document.
• There are two kinds of parameters for \texttt{show}:
  – \texttt{identifier <- exp} fills in a hole with some text; and
  – \texttt{identifier -> identifier} \texttt{reads the value of an input field}. 
exp : intconst
     | "true" | false
     | string
     | identifier
     | exp "=" exp | exp ">" exp
     | exp "and" exp | exp "or" exp
     | "not" exp
     | exp "+" exp | exp "-" exp | exp "*" exp
     | exp "/" exp | exp "%" exp
     | exp "++" exp
     | exp "(" exp ".." exp ")"
     | "|" exp "|"
     | "random" "(" exp ")"
     | "system" "(" exp ")"
     | "(" exp ")"
For You To Do

• Consider these questions about `show`:
  – what happens if not all holes are filled in?
  – what happens if not all input fields are read?
  – what happens if a hole is filled in with an integer?
  – what happens if a hole is filled in twice?

• There are many similar questions about the rest of the WIG language.
  – You will need to resolve them to implement your compiler.
Session Execution

Client

Server

browse

init

show

show

show

stop

browse

browse

browse
CGI-thread Realization

---

Client

- init
- save local state and stop
- browse and submit form
- restore local state and resume
- save local state and stop
- browse and submit form
- restore local state and resume
- save local state and stop
- browse and submit form
- restore local state and resume
- stop

Server

---

CIS 706 Translators I
Security Issues

• Tampering with service state
  – Keep all state on the server side

• Hijacking a session
  – Use randomly generated keys as session id

• Saving the state of a thread
  – Server makes control state explicit
An Example WIG Service

service {
  global x: int = 87;

  document Add {
    How much do you want to add?
    <input name="addition" type="text" size=4>
  }

  document Total { The total is now <var name="total">. }

  session Contribute {
    local i: int;
    show Add[addition->i];
    take x { x:=x+i };
    show Total[total<-x]
  }
}

CIS 706 Translators I
#include "runwig.h"
char *sessionid, *sessionkey, *password;

void output_Add() { ... }
void output_Total(char *var_total) { ... }

int PC, tempInt;
char *tempText;
int local_tiny_Contribute_i;

void save_Contribute_state(char *state,int pc) {
 FILE *f;
f = fopen(state,"w");
fprintf(f,"%i\n",pc);
fprintf(f,"%i\n",local_tiny_Contribute_i);
fclose(f);
}
void output_Total(char *var_total) {
    printf("Content-type: text/html\n\n");
    printf("<form method="POST" ");
    printf("action="/cgi-mis/tiny?%s">\n", sessionid);
    printf("The\n");
    printf("total\n");
    printf("is\n");
    printf("now\n");
    printf("%s\n", var_total);
    printf(".\n");
    printf("<p><input name="submit" 
        type="submit" value="continue">"));
}

CIS 706 Translators I
void main() {
    sessionid = getenv("QUERY_STRING");

    parseFields();
    if (strcmp(sessionid,"Contribute")==0) goto start_Contribute;
    if (strncmp(sessionid,"Contribute",10)==0) goto restart_Contribute;
    illegalRequest(sessionid);

    .
    .
    .
}
Contribute

session Contribute {
    local i: int;
    show Add[addition->i];
    take x { x:=x+i };
    show Total[total<-x]
}

start_Contribute:
local_tiny_Contribute_i = 0;
sessionId = randomString("Contribute",20);
sessionkey = sessionId+11;
save_Contribute_state(sessionkey,1);
output_Add();
exit(0);
session Contribute {
    local i: int;
    show Add[addition->i];
    take x { x:=x+i };
    show Total[total<-x]
}

Contribute_2:
lock("global_tiny_x");
tempInt = getGlobalInt("global_tiny_x") + local_tiny_Contribute_i;
putGlobalInt("global_tiny_x",tempInt);
unlock();
save_Contribute_state(sessionkey,3);
output_Total(itoa(getGlobalInt("global_tiny_x")));
exit(0);
session Contribute {
    local i: int;
    show Add[addition->i];
    take x { x:=x+i };
    show Total[total<-x]
}

Contribute_3:
terminateSession(sessionkey,NULL);
restart_Contribute:
sessionkey = sessionid+11;
if (freopen(sessionkey,"r",stdin)==NULL) illegalSession(sessionkey);
scanf("%i\n",&PC);
scanf("%i\n",&local_tiny_Contribute_i);
if (PC==1) goto Contribute_1;
if (PC==2) goto Contribute_2;
if (PC==3) goto Contribute_3;
}